Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds[1]

Wendy G. Lehnert
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
413-545-3639

Running Head: Symbolic/Subsymbolic Sentence Analysis

0.1 Introduction

Despite some recent speculations about the changing methodological styles in artificial intelligence, it is still possible to learn something by writing exploratory computer programs. This is especially true when a new processing technique or synthesis of different techniques is being attempted for the first time. In this spirit, we have designed and implemented a conceptual sentence analyzer, CIRCUS, which integrates three distinct information processing architectures.

CIRCUS uses (1) a stack-oriented control for syntactic predictions, (2) a marker-passing design for predictive preference semantics, and (3) numerical relaxation with lateral inhibition for data-driven preference semantics. Semantic preferences are therefore realized by two distinct mechanisms: we use marker-passing for the preferences associated with predictive semantics, and we use numerical relaxation for the preferences associated with data-driven semantics. This makes CIRCUS a blend of symbolic processing techniques (for 1 and 2) and subsymbolic processing techniques (for 3).

The distinction between predictive semantics and data-driven semantics is not a new one. For the sentence analyzers designed by the "Yale School", this distinction is usually described as the difference between *predictive* or *top-down slot-filling* versus *bottom-up slot insertion* (Riesbeck & Schank, 1976; Birnbaum & Selfridge, 1981; Dyer, 1983; Cullingford, 1986; Lytinen, 1984). Semantically-oriented sentence analyzers that have evolved from other traditions also acknowledge that the problem of filling existing slots within a conceptual frame is distinct from the problem of creating and inserting new slots into a frame that

[1]Acknowledgements: This research was supported by the Advanced Research Projects Agency of the Department of Defense, monitored by the Office of Naval Research under contract #N00014-87-K-0238, the Office of Naval Research, under a University Research Initiative Grant, Contract #N00014-86-K-0764 and NSF Presidential Young Investigators Award NSFIST-8351863.

does not predict such slots. For example, recent work in preference semantics differentiates noun and verb case information from preposition case information (Wilks, Huang & Fass, 1985).

To our knowledge, CIRCUS is the first system to characterize predictive semantics as a purely symbolic process while data-driven semantics is characterized primarily as a subsymbolic process. In this paper we will discuss some example sentences that illustrate the predictive/data-driven distinction and show how CIRCUS handles them. We further contend that any sentence analyzer which does not make a predictive/data-driven distinction must be either finessing a large class of problems, or trying to stretch a single processing mechanism farther than it can reasonably be expected to go.

When viewed in terms of linguistic models that are not process-oriented, we will see that our predictive/data-driven distinction does not carve up the world along the same lines as a linguist might. For example, linguists often view the problem of prepositional phrase attachment (pp-attachment) as a problem that can be resolved in a purely syntactic manner (Frazier & Fodor, 1979). When semantic considerations are introduced, they are usually based on lexical preferences from verbs (Ford, Bresnan & Kaplan, 1981). In CIRCUS we will see that some pp-attachments are resolved by predictive semantics while others require data-driven semantics. In other words, some pp-attachment problems can be solved by standard symbolic methods, while others are best addressed using subsymbolic techniques.

While we cannot hope to give a full technical description of CIRCUS in this paper, we will focus on those aspects of CIRCUS which are most interesting from the perspective of "high-level connectionism," and hope that our broad description is sufficient to convey a general sense of the overall system design.

0.2 Syntactic Processing

Over the years there has been much confusion about the status of syntactic knowledge within conceptual sentence analyzers. Although conceptual sentence analyzers do not produce syntactic parse trees, it is an overstatement to say that these systems never use syntax. What they don't use is a syntactic grammar, but it is nevertheless useful to recognize simple syntactic constituents like nouns and verbs, etc. The important claim is that syntactic knowledge must be instrumental to semantic interpretation - not just a means for identifying syntactic structure alone.

CIRCUS is consistent with this tradition although it is somewhat more principled about its organization of syntactic constraints. All syntactic predictions associated with dictionary items are isolated in the stack-oriented architecture of CIRCUS, and therefore kept separate from the processes that consider semantic constraints. This makes it easier to untangle the interactions between syntax and semantics, and it gives us an obvious advantage in constructing dictionary definitions.

Although we will not go into the details here, the syntactic processing in CIRCUS is well documented and available in a popular AI textbook (Schank & Riesbeck, 1981) where it is described as McELI, a micro version of a conceptual sentence analyzer designed by Chris Riesbeck (Riesbeck, 1975). Ironically, we use McELI inside CIRCUS to do nothing but recognize syntactic sentence constituents, and store appropriate sentence fragments inside global buffers that keep track of constituents like the subject of the sentence, direct and indirect objects, prepositional phrases, and so forth. The buffers are restricted to simple syntactic structures with a strongly "local" sense of the sentence: larger constituents like clauses are not recognized using explicit syntactic buffers.

No attempt is made to resolve all syntactic ambiguities as soon as they are encountered. Syntactic ambiguities will be handled incrementally when the marker passing algorithm interacts with McELI and contributes the constraints it derives from predictive preference semantics. If we see "John gave Mary ..." we will retain Mary as a possible binding for both the direct object and the indirect object until more information is made available to us. This wait-and-see strategy is necessary since many sentences can resolve themselves in more than one way ("John gave Mary a kiss" vs. "John gave Mary to the sheik"). The marker passing algorithm described in the next section will generate a semantic preference which gives us a bias for Mary as an indirect object, but the syntactic processing of McELI will not relinquish its multiple interpretations unless the ambiguity can be resolved on the basis of syntactic constraints alone (as would be the case for "John gave Mary to the sheik").

To illustrate the syntactic predictions used by McELI, consider the verb "to give." In its active form, we can expect this verb to predict (1) a direct object, (2) a direct object followed by a prepositional phrase using the preposition "to," (3) a prepositional phrase with "to" followed by a direct object, or (4) an indirect object followed by a direct object. The following sentences illustrate each possibility:

(1) John gave a book.
(2) John gave a book to Mary.
(3) John gave to Mary a book.
(4) John gave Mary a book.

These predictions are not unique to the verb "to give," nor do they apply to all verbs in general. Using the request packet mechanism of McELI, it is easy to encode these expectations. When an ambiguity can be resolved on the basis of syntax alone, McELI can handle the resolution without additional help. For example, in processing (2), McELI will initially fill both the direct object buffer and the indirect object buffer with the noun phrase "a book." Three predictions are active at that point. One prediction expects to see a prepositional phrase next (as in 2), and another expects to see a second noun phrase (as in 4). If the first situation is encountered, we empty the indirect object buffer as soon as the prepositional phrase is recognized. If the second situation arises, we overwrite the direct object buffer when the second noun phrase is found. If the sentence terminates (as in 1), we empty the indirect object buffer as before. In any case, the momentary ambiguity is resolved by combining limited nondeterminancy with a wait-and-see strategy.

Because the syntactic constituents recognized by McELI are immediately given to the marker passing algorithm to augment purely semantic constraints and preferences, we cannot describe the role they play without talking about marker passing in CIRCUS. To conclude this section, we will merely point out that while McELI is a severely limited "toy" when viewed as a complete conceptual sentence analyzer, we have found McELI to be quite capable when regarded as a subsystem that is expected to do nothing more than recognize simple syntactic constituents in a deterministic manner. The stack-oriented architecture of McELI is not overwhelmed by this task, and it is easy to encode appropriate dictionary definitions for McELI that describe the simple syntactic predictions needed by CIRCUS.

0.3 Predictive Preference Semantics

The predictive semantics module (PSM) in CIRCUS is responsible for filling slots in pre-defined semantic case frames. CIRCUS makes no commitment to a particular style of semantic representation, although we favor the "deep" semantic case frames of the sort found in conceptual dependency (Schank, 1975) as opposed to lexical case frames (Simmons, 1984). The more complex case frames are usually associated with verbs, but simple case frames can also be associated with nouns. Semantic dictionary entries for lexical items must list multiple frames to differentiate multiple word senses, although specific word senses can be effectively ignored by excluding them from the dictionary. When two or more word senses are present to compete with one another, preference semantics must try to resolve this competition and determine which word sense is most likely in the context of the given sentence.

PSM interacts with McELI each time a new syntactic constituent is recognized by McELI. Control is passed from McELI to PSM and then back again to McELI. While PSM is active, two primary tasks are accomplished: (1) network construction, and (2) marker passing. If a case frame satisfies certain instantiation criteria, PSM will also "freeze" that case frame with its assigned slot fillers for future access as a part of the conceptual meaning representation CIRCUS derives for the sentence. We will now talk about each of these tasks in more detail.

0.3.1 Network Construction

Each network constructed by PSM consists of three types of nodes: (1) syntactic nodes, (2) concept nodes, and (3) semantic feature nodes. Different types of information are passed between different node combinations, and different links are used to channel different kinds of information. The basic goal of the network is to activate concept nodes which will contribute to the conceptual interpretation of the sentence. The network is constructed with the following restrictions:

1. syntactic nodes can only be connected to semantic feature nodes
   (1a) these may be connected by a *soft constraint link*, or
   (1b) these may be connected by a *hard constraint link*
2. concept nodes can only be connected to semantic feature nodes
   (2a) these may be connected by a *slot-filling link*, or
   (2b) these may be connected by an *enablement link*
3. multiple syntactic nodes can be connected to a given semantic feature node
4. multiple semantic feature nodes can be connected to a given syntactic node
5. multiple semantic feature nodes can be connected to a given concept node
6. multiple concept nodes can be connected to a given semantic feature node

Definitions describing specific concept nodes and semantic feature nodes must be provided to PSM before sentence analysis begins. All semantic dictionary entries are defined in terms of these two node types. Concept nodes are added to the network when a lexical item in the sentence contains a dictionary definition for a concept node, although the presence of such a dictionary entry is not a sufficient condition by itself for network expansion. Semantic feature nodes are added to the network whenever a newly activated concept node references them in its definition. Syntactic nodes are added to the network whenever McELI assigns a sentence fragment to a constituent buffer.

To get a more concrete sense of what is going on during network construction, let's take a look at a specific example. We will take as our example the network constructed by PSM in order to process "John gave Mary to the sheik." After we have described the network construction for this sentence, we will then discuss the marker passing algorithm that uses this network.

The complete network generated by "John gave Mary to the sheik" appears in figure 1. To see how this is incrementally constructed, we'll describe each partial construction as control moves back and forth between McELI and PSM.

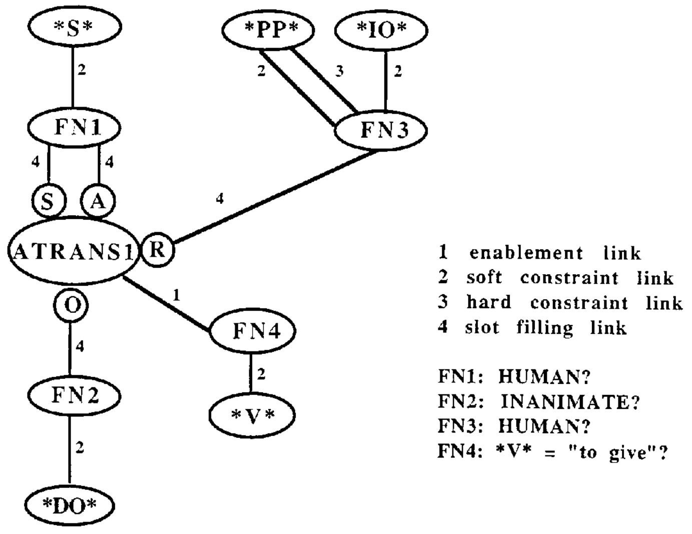

Figure 1: The PSM Network for "to give".

At this point McELI hypothesizes a subject for its sentence and places "John" in the *S* buffer. When control passes to PSM, we construct a single syntactic node marked *S* in figure 1. The value associated with this node is the sentence fragment "John." PSM also consults its semantic dictionary to see if there are any concept node entries under John, but finds nothing.

[2] John gave ...

McELI places "gave" in the *V* buffer, and a syntactic node *V* is created for the network. This time, when PSM consults its semantic dictionary, it finds a single concept node under the verb "gave."[2] This concept node describes an ATRANS event[3] which appears as ATRANS1 in figure 1. The concept node definition ATRANS1 is designed to drive the bulk of our network construction from here on through the end of the sentence. For now, a semantic feature node FN1 is added to the network with a soft constraint link between *S* and FN1, and slot-filling links between FN1 and ATRANS1. We also add a second semantic feature node FN4, with a soft constraint link between *V* and FN4, and an enablement link between ATRANS1 and FN4. We will discuss the utility of these semantic feature nodes when we describe the marker passing algorithm.

[3] John gave Mary ...

McELI now passes two buffers to PSM, both of which contain the sentence fragment "Mary." The *DO* buffer hypothesizes that Mary may be a direct object, and the *IO* buffer hypothesizes that Mary may be an indirect object. PSM creates two corresponding syntactic nodes, marked *DO* and *IO* in figure 1. PSM then consults the definition for ATRANS1 and adds the semantic nodes FN2 and FN3 to the network, using a soft constraint link to join *DO* to FN2, a slot-filling link to join FN2 to ATRANS1, a soft constraint link to join *IO* to FN3, and a slot-filling link to join FN3 to ATRANS1. Finally, PSM consults its semantic dictionary to see if there is a concept node definition associated with Mary, but it finds none.

[4] John gave Mary to the sheik.

McELI now passes a *PP* buffer to PSM which contains the sentence fragment "to the sheik." PSM builds a syntactic node marked *PP* in figure 1. PSM also consults the definition for ATRANS1 and determines that *PP* should be joined to FN3 using both a soft constraint link and a hard constraint link. PSM also consults its semantic dictionary to see if there is a concept node definition associated with "sheik" but it finds nothing.

Each time control passes to PSM, we complete whatever network construction is needed, and then execute the marker passing algorithm. We will now describe marker passing in PSM.

[2]This concept node definition should really appear under an entry for the infinitive "to give" but we don't have any morphology routines hooked up to CIRCUS at the present time.

[3]ATRANS is a primitive act in conceptual dependency (Schank, 1975).

0.3.2 Marker Passing

In order for the marker passing algorithm to run, we must have at least one concept node present in the network. For our example sentence, this first occurs after we've processed "John gave ..." As described above, the network at that time consisted of *S*, *V*, FN1, FN4, and ATRANS1. In general, information flows from the syntactic nodes to the semantic feature nodes, and from the semantic feature nodes to the concept nodes. We can think of this spreading activation as a parallel process since there are no order effects or time-sensitive dependencies during marker passing.

When a semantic feature node completes a path from a syntactic node to a concept node, the semantic feature node acts as a semantic constraint checker to see if the concept frame associated with the concept node should fill a slot, or maybe remove itself from the network altogether. Slot filling links are used to channel potential slot fillers, and enablement links are used to sustain concept nodes in the network.

In our example, FN1 is designed to check the head of the subject noun phrase found in *S*, and determine whether or not this noun refers to a human. This determination is made in accordance with whatever memory model is available to CIRCUS, a problem we will discuss later in section 5. For now, we will simply assume that a feature node can pass a value of 1 or 0, depending on whether its test returns true or false. We will refer to these numbers as semantic preference values. In this case, John tests out as a human, and FN1 will pass both the slot filler, John, to ATRANS1 along with its semantic preference value of 1. We should note that each slot-filling link into ATRANS1 also specifies the targeted slot for our slot-filler. In this case, we have a pair of slots specifying the actor and the source for the ATRANS frame (the small letters circled at the end of each slot-filling link indicate which slot is being targeted by that link).

The other semantic feature node, FN4, is joined to ATRANS1 by an enablement link which operates to confirm the viability of the concept node as the sentence progresses. In our case, FN4 simply checks that the contents of *V* is a form of the verb "to give," a condition that will test true as long as "gave" is the only verb we encounter in the sentence. In general, enablement conditions are used to dismiss concept frames at any time after they are triggered by dictionary look-up, usually after additional constraints within the sentence appear and prohibit the possibility of a given word sense.

At this point in the sentence (John gave ...) the marker passing algorithm passes a confirmation on the enablement condition to ATRANS1, and semantic preference values yielding a sum of 2 to the ATRANS concept frame on the basis of John as both an actor and source for the ATRANS. We therefore keep ATRANS1 in the network, and assign it an activation level of 2/4 (we divide by 4 because there are 4 variable slots in the ATRANS frame: actor, object, source, and recipient).

```
The following concept has activation level .5

                event = ATRANS
                actor = JOHN
                source = JOHN
```

By the time we reach "John gave Mary ..." we have added two more syntactic nodes, *DO* and *IO*, and two more semantic feature nodes, FN2 and FN3. FN2 checks to see if the direct object is an inanimate object, and FN3 checks to see if the indirect object is a human. Since Mary is bound to both *DO* and *IO*, FN2 fails and FN3 succeeds. FN2 then passes Mary to ATRANS1 as a candidate for the object slot with a preference value of 0, and FN3 passes Mary to ATRANS1 as a candidate for the recipient slot with a preference value of 1. Because these preference values are interpreted as soft constraints (the semantic feature nodes received their potential slot fillers via soft constraint links), the ATRANS frame receives both slot fillers, and adds Mary to the ATRANS frame as both an object and a recipient. However, PSM marks the object slot with a flag to indicate that this slot filler violates a semantic preference, and the activation level rises only to .75, indicating that all is not well as far as the frame instantiation is concerned.

```
The following concept has activation level .75

        event = ATRANS
        actor = JOHN
        object = MARY <<< semantic violation
        source = JOHN
        recipient = MARY
```

When the sentence is completed (John gave Mary to the sheik) we have the complete network at our disposal, and the marker passing algorithm completes its frame instantiation using ATRANS1. The only addition to the network is a syntactic node, *PP*, and this node is joined to FN3 via a soft constraint link and a hard constraint link. Given a prepositional phrase to consider, FN3 can now pass new information on to ATRANS1. First recall that McELI will now empty the *IO* buffer that feeds the syntactic node so that *IO* no longer points to Mary. The only channel feeding information to FN3 are now the links joining FN3 to *PP*. The hard constraint link requires that *PP* hold a prepositional phrase using the preposition "to." If this condition is not met, FN3 will not pass any potential slot fillers on to ATRANS1 on the basis of *PP*. Since we pass this test, we then consider the soft constraint link, which asks only that the object of the prepositional phrase be human. As a soft constraint, this test could fail without blocking the slot filler (just as we saw for FN2 above), but this time, the sheik qualifies and is passed to ATRANS1 as a slot filler for the recipient slot with a preference value of 1.

```
The following concept has activation level .75

          event = ATRANS
          actor = JOHN
          object = MARY <<< semantic violation
          source = JOHN
          recipient = SHEIK
```

At this point, we are done with our sentence and we can take our semantic meaning representation from ATRANS1, the only concept node in the network. We have an ATRANS event with actor = John, object = Mary, source = John, and recipient = the sheik. Note that the activation level is still only .75, since the slot filler for the ATRANS object still fails to meet the semantic preference associated with that slot (Mary is not an inanimate object). When PSM returns this instantiation of the ATRANS frame as the conceptual representation for our sentence, it flags this slot filler as one which violated a semantic preference.

0.3.3 Resolving Competing Word Senses using Predictive Semantics

Before we move on to data-driven semantics, we should point out that some word sense ambiguities can be resolved on the basis of predictive semantics alone. For example, suppose we wanted to understand the sentence "John gave Mary a kiss" as well as "John gave Mary to the sheik." Now we no longer want to interpret the event as an ATRANS event. (No transfer of possession occurs here as it did with the sheik). For lack of a better decomposition, we'll simply characterize it as a kissing event.

However, both sentences start out with the same verb, and we can't reject ATRANS1 as a valid concept node on the basis of any hard enablements. In fact, ATRANS1 will receive the same amount of activation that it had for "John gave Mary to the sheik" since only the object of the ATRANS frame violates its semantic preference for an inanimate object. Indeed, if our dictionary recognized the possibility of a chocolate kiss, we could give ATRANS1 an activation level of 1 and be perfectly confident about this interpretation. Barring, for the moment, the idea of a candy kiss[4], we can only reject an ATRANS1 interpretation with activation level = .75 if some competing concept node can come up with a higher activation level.

Figure 2 shows how a competing concept node can be brought in to establish an interpretation of the sentence based on a kissing event. In this case, the KISS1 concept node resides in the semantic dictionary under the noun definition for "kiss." This is the appropriate place to index a possible kissing event, since we don't want to predict a possible kiss every time we see the verb "to give."

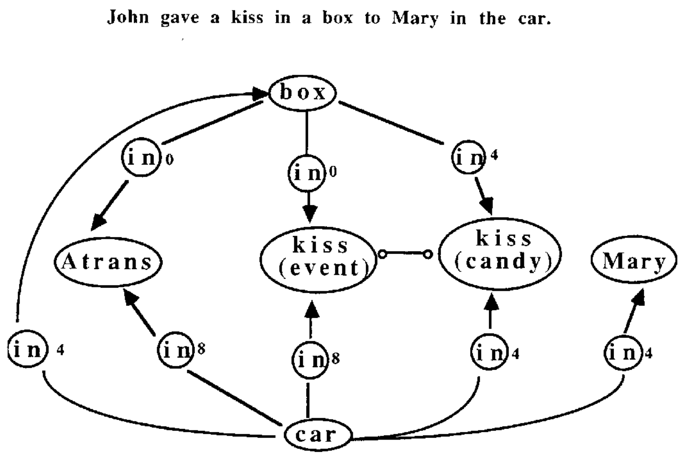

Figure 2: Data-Driven PP-Attachment.

Note that KISS1 introduces only one new semantic feature node, FN5. The rest of its links hook up to nodes already present in the the network because of ATRANS1. KISS1 has two enablement links from FN4 and FN5, and two slot-filling links from FN1 and FN3. It shares the same hard and soft constraints used by ATRANS1 to fill its actor and recipient slots, so it can tap into those parts of the network without additional effort. For enablements, it simply requires

[4] We will come back to this problem of lexical ambiguity in section 4.1.

the verb "to give" (FN4) and a direct object which is an event (FN5). Since the first enablement is shared with ATRANS1, we can once again tap into the existing network structures without further construction. Only FN5 and the two links connecting it need to be added when KISS1 is added to the network.

Now the marker passing algorithm passes slot fillers to both ATRANS1 and KISS1, in which case ATRANS1 arrives at an activation level of 3/4 (we have a preference violation on the object slot), and KISS1 determines that John can be an actor and Mary can be a recipient without any preference violations, yielding a concept node activation level of 1. A simple comparison of these activation levels will then tell us which of the surviving concept nodes holds the best interpretation of the sentence.

0.3.4 Resolving Competing Predictions Across Nested Case Frames

In the last section we saw how multiple concept nodes could compete with one another to settle a word sense ambiguity. In such competitive situations we want to see one concept node win so only one can participate in the final interpretation of the sentence. But it is also possible for multiple concept nodes to create nested case frame instantiations in a final meaning representation. Then we want to see multiple concept nodes cooperate with one another and stay out of each other's way as much as possible. To see how these cooperative situations arise, consider the following sentences:

[S1] John gave Mary the key to the city.

[S2] John gave the key to the city to Mary.

Here we have the ATRANS1 concept node attempting to instantiate an ATRANS frame, while a second concept node associated with the concept of a key is also trying to fill a simple frame describing a key and the object the key opens. To make matters even worse, we'll assume that the KEY concept node has a semantic preference on its single slot (the opens-slot) that the slot filler be something which has a lock, a condition not met by cities. For both S1 and S2, we want to see the KEY case frame instantiated despite the fact that its slot filler does not satisfy the semantic preference associated with that frame. Both S1 and S2 should produce the following meaning representation:

```
event = ATRANS
actor = JOHN
object =
         header = KEY
         opens = CITY <<< semantic violation
source = JOHN
recipient = MARY
```

We will use the ATRANS1 concept node described before, and we can easily define KEY to fill its opens-slot on the basis of a prepositional phrase with the preposition "to" (a hard constraint), and a head noun which can have a lock (a soft constraint). In S1, the ATRANS frame will be fully instantiated as soon as we get to "the key." We need only complete the KEY frame instantiation when the final prepositional phrase arrives.

To recognize the nested case frame relation, we need to add one new link type to the network grammar described in section 3.1. We will now allow a case-frame value link to connect syntactic nodes to concept nodes whenever a noun sense triggers a concept node definition. Figure 3 shows the PSM network needed to handle S1 and S2.

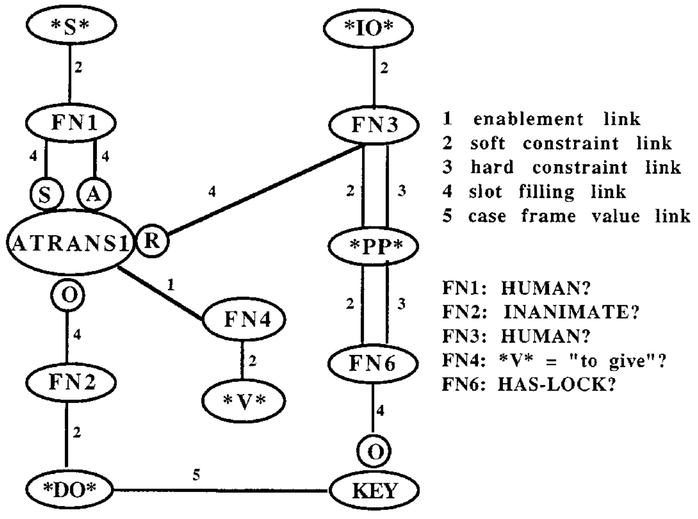

Figure 3: A PSM Network for Nested Case Frames.

A trickier situation arises with S2. Now we have to watch out for the fact the the *PP* buffer is going to get overwritten by McELI when the second prepositional phrase is encountered. This is fine for the ATRANS frame, which should be picking up Mary as its recipient anyway. But it is not so good for the KEY frame, which will lose its pathway to the city and fail to pick up the correct slot filler.

PSM handles this by putting a time limitation on each concept frame to make sure that slot filling constituents which are too far removed from a potential concept node are not considered by that concept node. Distance is measured by a clock that ticks off the syntactic constituents recognized by McELI. For simple concept nodes triggered by nouns, this heuristic is very effective. For example, the KEY frame should never look past the syntactic constituent immediately following the noun "key."

When the time limitation for a concept node is up, a mechanism freezes the current frame instantiation, and procedures that access the concept node from that time on will only see the frozen case frame instantiation no matter what else is happening in the PSM network. In this way, it is possible to overwrite the information within a syntactic node and still retain old information within a case frame instantiation as needed.

Now let's look at another trouble maker:

[S3] John gave the key to the city.

In this sentence, we have both the ATRANS case frame and the KEY case frame actively competing for the second prepositional phrase, but neither of them finds their semantic preference satisfied. Who wins? Normally, we look at the semantic preferences to resolve this sort of competition. If John gave the key to the car, we'd want to see the KEY frame win. If John gave the key to Mary, we'd want to see the ATRANS frame win. Because CIRCUS keeps track of slot fillers that violate preferences, it is possible to resolve situations where two frames grab the same constituent, but only one is semantically satisfied. In cases like S3, we allow CIRCUS to remain uncertain about the outcome. Without a strong semantic preference to guide us, S3 remains semantically ambiguous and no further attempt is made by CIRCUS to resolve the problem.

Note that a "key to the city" is just as anomalous as "a key to the chair" if we stick to the KEY frame definition given above. If we had another word sense for key which encoded the ceremonial notion of a symbolic key, we could set up a competition between the multiple concept nodes associated with keys, and resolve the problem that way. This explains why people reading this sentence do have a preference which would swing their interpretation toward some ceremonial understanding of the event in question. CIRCUS can arrive at the same conclusion, but only if we give it adequate knowledge about multiple word senses.

Notice that each of these examples described a problem with prepositional phrase attachment. When two concept frames compete for the same prepositional phrase, we are looking at competing attachment points for the prepositional phrase. The ambiguity in S3 arises from the fact that "to the city" might be attached to either the verb "gave" or the noun "key." In these instances, CIRCUS must rely on predictive semantics alone to make judgment about the best attachment. Given an adequate dictionary of multiple word sense definitions and appropriate semantic preferences within those case frame definitions, we can expect predictive semantics to suffice for these attachment problems. However, other attachment problems associated with prepositional phrases cannot be resolved on the basis of predictive semantics alone. We will consider these problems in the next section.

0.4 Data-Driven Preference Semantics

Each of the following sentences contains a prepositional phrase that cannot be interpreted on the basis of predictive semantics alone. Any verb or noun can act as the attachment point for a prepositional phrase describing a locational setting (in the car or in a box). If we treated these modifiers in terms of predicted slots, we would have to duplicate a very large number of predictions for every case frame in our dictionary.

[S4] John **gave** the key to Mary *in the car.*

[S5] John **gave** Mary the key to the car *in the car.*

[S6] John **gave** the key to the city to Mary *in the car.*

[S7] John **gave** the key to the car to Mary *in the city.*

[S8] John gave the **key** *in a box.*

[S9] John gave the **key** to the city *in a box* to Mary.

[S10] John gave the **key** to the city to Mary *in a box.*

[S11] John gave the **key** to the car to Mary *in a box.*

Because these prepositional phrases operate with such sweeping generality, it makes sense to interpret them in a more bottom-up fashion, dealing with each instance only as it appears without top-down expectations. The final prepositional phrase in each of S4-S7 attaches to the verb, crossing 2 or 3 intermediate constituents to make the attachment. In S8-S11 we see a prepositional phrase attaching to a noun, with 0, 1, or 2 intermediate constituents intervening. Notice that "in a box" could conceivably attach to "gave" if you are willing to imagine a box large enough to contain John and Mary. But if your prototypical box is somewhat smaller, you will prefer a descriptor that limits its scope to the location of the key.

Although these examples all illustrate attachment points that coincide with conceptual frames (ATRANS1 and KEY), any noun can qualify as a possible attachment point regardless of whether or not it carries a case frame definition (John gave Mary a necklace in a box). In general, any noun or verb preceding a prepositional phrase can operate as a potential attachment point for that prepositional phrase, as long as we don't cross certain clause boundaries (like the main verb). To determine the best attachment point, some mechanism must be invoked that can weigh all the different possibilities and come up with the best interpretation. The constraints that inform this decision are primarily semantic constraints, although we will discuss a syntactic constraint for prepositional phrase attachment in section 4.3.

0.4.1 The Lateral Inhibition Network

To resolve the problem of bottom-up prepositional phrase attachment, we will first assume that any prepositional phrase claimed by a case frame slot prediction does not require further analysis, even if the slot filler violates a soft constraint as described in the previous section. The network relaxation mechanism we are about to describe only applies to prepositional phrases that are left uninterpreted by predictive semantics.

When such a prepositional phrase is encountered, we construct a network whose nodes are joined by both activation links and inhibitory links. To illustrate the network construction, consider the following sentence:

[S12] John gave a kiss in a box to Mary in the car.

Here we have two prepositional phrases that are unresolved by predictive semantics: "in a box," and "in the car." To make matters even more interesting, we will supply our dictionary with two word senses for the lexical item "kiss." KISS_E will be an event involving an actor and a recipient. KISS_C will be a piece of chocolate candy wrapped in tinfoil.

Each node in the relaxation network corresponds to (1) an attachment point, (2) a head noun in a prepositional phrase, or (3) a preposition. In fact, we have distinct nodes for each separate word sense associated with (1), (2), or (3), but our discussion of S12 will only incorporate multiple word senses for the kiss (two instances of attachment points). Moreover, copies of prepositional word sense nodes will be made for each possible attachment point associated with a given prepositional phrase. We will therefore refer to nodes of type (3) as relational nodes, since they each describe a relational interpretation between the head noun and a potential attachment point. It follows that if multiple word senses are associated with a given attachment point, separate relational nodes are created for each targeted word sense as well (KISS$_E$ and KISS$_C$ illustrate this situation).

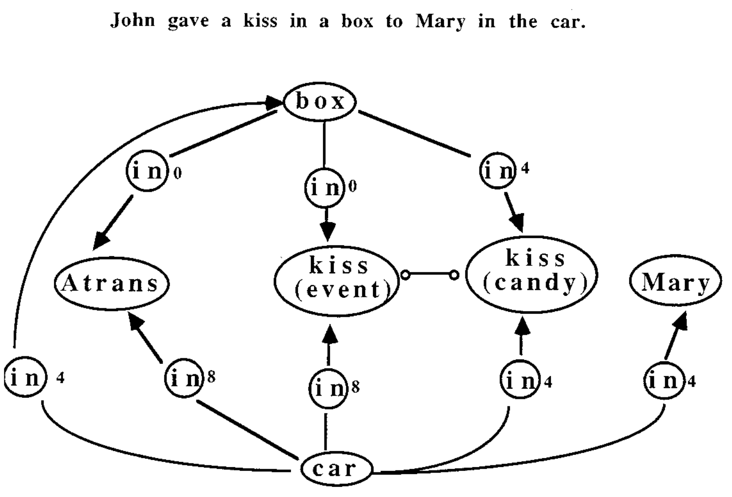

Figure 4: Data-Driven PP-Attachment.

Figure 4 shows the relaxation network associated with S12. We can see from this diagram that "in a box" has three possible attachments associated with it: one pointing to ATRANS1, one pointing to KISS$_E$, and one pointing to KISS$_C$. "In the car" has five possible attachment points: the box, ATRANS1, KISS$_E$, KISS$_C$, and Mary. Notice that this semantic interpretation of the attachment problem does not correspond perfectly to the syntactic notion of prepositional phrase attachment. While KISS$_E$ and KISS$_C$ are both triggered by the noun phrase "a kiss," KISS$_E$ serves to disambiguate the sense in which "gave" should be understood. Giving a kiss is not like giving a book. So an attachment to KISS$_E$ is really a syntactic attachment to the verb "gave" rather than the noun "kiss." However, we are not interested in building a syntactic parse tree: our only goal is to find the right case frame for a bottom-up slot insertion. Since our network is created with semantic entities rather than syntactic ones, the notion of a purely syntactic prepositional phrase attachment need not concern us. Syntactic attachments are rather beside the point, and they can be effectively bypassed without any trouble. We are treating prepositional phrase attachment as a strictly semantic problem with purely semantic solutions.

In figure 4 we have shown an inhibitory link between KISS_E and KISS_C, indicating that these two nodes are mutually exclusive and in competition with one another. All other links shown in figure 4 are activation links. In fact, there are more inhibitory links in this network than we have attempted to show in our figure. Each relational node associated with a given prepositional phrase must inhibit every other relational node created by the same prepositional phrase since all of these relational nodes signify competing interpretations for the given phrase. We therefore have 3 inhibitory links (not shown) connecting the 3 relational nodes associated with "in a box," and 10 inhibitory links (not shown) connecting the 5 relational nodes associated with "in the car." Once this relaxation network has been constructed, we can execute our relaxation algorithm.

Each node in the relaxation network is initialized with a value between 0 and 10. Only the relational nodes can be initialized with non-zero values. The initial value given to a relational node is determined by the available memory model and will be discussed in section 5. For now, we will assume that the following conditional is used to compute initialization values to describe the relation "X in Y":

```
IF X is an event and Y is bigger than a person,
THEN return 8,
ELSE IF X is not an event and X is not bigger than Y,
     THEN return 4,
     ELSE return 0.
```

Figure 4 shows how the relational nodes are initialized using the above rule. Having initialized the network nodes, we can now apply a simple relaxation algorithm (Feldman & Ballard, 1982) until the network stabilizes and each node settles down to a steady (or near-steady) value. The exact algorithm we use in CIRCUS is the same one we have used elsewhere (Lehnert, 1987a) so we will not bother to go into further detail about that here.

Once the network has stabilized, we can interpret the resulting node values to resolve three separate problems. One relational node should beat out the competition to attach each of the two prepositional phrases, and either KISS_E or KISS_C should win to resolve the lexical/conceptual ambiguity introduced by "a kiss".

Notice that the ambiguity of the kiss can only be resolved by understanding the constraints associated with "in a box." If this prepositional phrase can attach to KISS$_C$, but not KISS$_E$, then we will receive a preference for KISS$_C$ over KISS$_E$. If the box is big enough to allow either attachment, we will receive no preference for one sense of kiss over the other. "John gave a kiss to Mary in the car," leaves us much more undecided about the kiss ambiguity than S12 does.

Although it may not be obvious from the initial node values in figure 4, the relaxation algorithm will eventually stabilize with attachments that describe a piece of candy in a box, and an ATRANS event taking place in the car. The final output from CIRCUS will therefore be:

```
event =   ATRANS
actor =   JOHN
object =  KISS_C
          [relational-link = IN(BOX)]
source =  JOHN
recipient = MARY
[relational-link = IN(CAR)]
```

In the next section we will explain how results from the relaxation algorithm are used by PSM in order to produce this representation.

0.4.2 Interactions between Predictive and Data-Driven Semantics

Although the data-driven processing described in the last section is useful in resolving certain ambiguities associated with multiple word senses and prepositional phrase attachment, we must still coordinate these results with the preferences and results obtained by predictive semantics. A dynamic interaction between predictive semantics and data-driven semantics occurs if the relaxation algorithm is invoked after each syntactic constituent is recognized. As we saw earlier, control first passes to predictive semantics which produces its case frame instantiations and preferences without the benefit of any data-driven processing. Then the relaxation network is constructed on the basis of whatever we have seen so far, and the relaxation algorithm passes its results back to predictive semantics in case any data-driven problem resolutions can influence a case frame instantiation or competition between case frames.

There are three ways that data-driven semantics can influence and augment predictive semantics. These are:

(1) Bottom-up slot insertion
(2) Word sense preferences for predicted slot fillers
(3)Enabling conditions for predictive case frames

Bottom-up slot insertion is the simplest example of the data-driven contributions. When a prepositional phrase is not recognized by predictive semantics, data-driven semantics must attempt to find the best relational interpretation for the prepositional phrase. Once that relation has been identified, we simply insert a new slot inside the appropriate case frame which describes the relation in question. For example, if John gave Mary a book on Tuesday, we would insert the slot (relational-link = TIME (TUESDAY)) inside the ATRANS1 instantiation.

Although word sense preferences are addressed by the soft constraints of predictive semantics, we can also benefit from the preferences of data-driven semantics when multiple word senses are involved. In S12 we saw how KISS_C will beat out KISS_E as a result of the relaxation algorithm after "in the box" is encountered. Although ATRANS1 had a soft preference for KISS_C over KISS_E, we were not prepared to dismiss KISS_E on the basis of predictive semantics alone (remember that "John gave Mary a kiss in the car" remains ambiguous without the constraints from "in a box" to resolve the word sense of kiss). But once data-driven semantics has given us a strong preference for KISS_C over KISS_E, we can now go back to predictive semantics with the message that KISS_E should not be considered as a possible slot filler for any case frames that might have wanted it. In S12, this will suppress an interpretation where ATRANS1 allows KISS_E to operate as a possible object slot filler despite the preference violation that occurs when an event is pushed into the ATRANS object slot.

The final way in which data-driven results can influence predictive semantics is through the enabling conditions associated with predictive case frames. This is how the competition between KISS_E and ATRANS1 is actually resolved in S12. When the KISS_E word sense for "a kiss" is dismissed by the relaxation algorithm, the enabling condition for the KISS1 case frame will fail to be met since the direct object of "gave" can no longer describe an event. Once a case frame enablement fails, the case frame is removed from consideration by predictive semantics, leaving only surviving case frames as viable contenders for the final interpretation.

Because enabling conditions for predictive case frames may be sensitive to timing effects as we move through the sentence, this last form of interaction between data-driven semantics and predictive semantics makes it desirable to run the relaxation algorithm periodically as we move through the sentence. If we wait until the end of the sentence to bring in data-driven effects, there is a danger of missing some failed enabling condition which was detectable for only a limited time as we moved through the sentence (most likely because McELI overwrites its syntactic buffers). This suggests that the two processes must operate in a roughly parallel fashion, with frequent communication between them.

It is also important to note that in every interaction described here, information always flows from the data-driven component to the predictive component. We never see information from predictive semantics influencing the relaxation algorithm. For this reason, we could say that predictive semantics operates in a strictly top-down fashion, while data-driven semantics is truly bottom-up. Bottom-up information can influence top-down processing, but top-down processing cannot influence a process that is purely bottom-up. This suggests an important claim about the relationship between symbolic and subsymbolic processes which sounds quite plausible in general: Symbolic processes can be influenced by subsymbolic processes, but the converse does not hold. If we assume that symbolic processes are basically serial while subsymbolic processes are essentially parallel, we have a corollary to this claim: serial processes can be influenced by parallel processing, but the converse does not hold. It is interesting to note that experimental reaction time results on lexical access appear to be completely consistent with this general rule of serial/parallel interaction (Swinney, 1984).

0.4.3 The No-Crossing-of-Branches Rule

At this point we have outlined the basic mechanism of data-driven processing, but we have not argued for the necessity of network relaxation. If S12 were our only example of data-driven processing, it would seem that simple heuristics based on symbolic memory interactions should suffice. In this section we will present a more compelling argument for the necessity of relaxation algorithms in data-driven processing.

The most striking argument seems to arise from a syntactic constraint called the no-crossing-of-branches rule. This rule accounts for the dissonance that most people experience when reading the following sentence:

[S13] The man saw the woman with binoculars in a green dress.

Most people who read this sentence conclude that (1) the man was using binoculars and the woman was wearing a green dress, but the sentence is poorly worded, or (2) the woman was carrying binoculars and wearing a green dress. The first interpretation is based on semantic preferences which strongly suggest that the binoculars should be an instrument for seeing, and the woman must be the one wearing the green dress. Any discomfort with the wording of the sentence under this interpretation is due to the fact that this semantically valid interpretation violates a syntactic rule about pp-attachment.

The no-crossing-of-branches rule goes into effect when there are multiple prepositional phrases in a sentence. It says that no two attachments can cross each other in going from prepositional phrase to attachment point. (1) violates the no-crossing-of-branches rule because the semantically-preferred pp-attachments for this sentence do cross one another:

The man saw the woman with binoculars in a green dress.

The second reading (2) of S13 is syntactically more acceptable, but perhaps a bit bothersome semantically since the instrumentality of binoculars to the act of seeing is so strong that we are reluctant to dismiss this attachment option. Unfortunately, the only way that the man can be using the binoculars without violating the no-crossing-of-branches rule is if the man or the binoculars wear the dress. Few readers are willing to entertain these possibilities as acceptable interpretations for the sentence.

As we can see, the no-crossing-of-branches rule is a soft constraint. It can be violated if semantic constraints are strong enough to force an interpretation that renders the sentence poorly worded. But in the absence of any such compelling constraints, the no-crossing-of-branches rule normally contributes to a productive unraveling of multiple pp-attachments.

It has been argued that the no-crossing of branches constraint can only be realized by a sentence analyzer which "performs complete syntactic processing of its input" (Tait, 1983). The reasoning here starts with the observation that no-crossing is a non-local constraint because a prepositional phrase can cross over an arbitrarily large segment of the sentence to reach its intended attachment point. It follows that any process which can detect a no-crossing violation must access a global representation of the sentence structure (e.g. a syntactic parse tree) in order to exploit the no-crossing constraint. Semantically-oriented analyzers which "deal only with local word-order effects" [op. cit.] cannot hope to apply no-crossing to pp-attachment routines because the no-crossing rule is properly concerned with global word order effects.

Although the claim that localized word orders cannot handle essentially global problems is quite correct, it is not true that global constraints like the no-crossing rule require complete syntactic processing. In CIRCUS we have implemented the no-crossing-of-branches rule without any need for a syntactic parse tree or syntactic structures beyond those described in section 2 (which are highly localized). All we need is the lateral inhibition network described in section 4.1, and a time stamp on each syntactic constituent associated with prepositional phrases and attachment points. If we maintain pointers from the nodes in the lateral inhibition network to the time-stamped syntactic constituents underlying them, we can use this simple time stamp to recognize no-crossing violations.

The time stamp will give us an adequate sense of the global relationships we need to consider, and no one would characterize a set of time-stamped sentence constituents as much of a syntactic parse tree.

To see how the no-crossing rule is implemented in CIRCUS, consider figure 5 where we have drawn a picture of the lateral inhibition network needed to process S13. Here we see two competing word senses for the preposition "with" and one word sense for the preposition "in." (A serious dictionary would provide us with many more word senses for these prepositions but our picture would only be muddied up by more attachment nodes). Each of the two word senses for "with" generates two attachment nodes and the single word sense for "in" generates three attachment nodes. (Note that figure 5 does not show all of the attachment nodes associated with "with." Once again, we are only trying to keep the picture simple.)

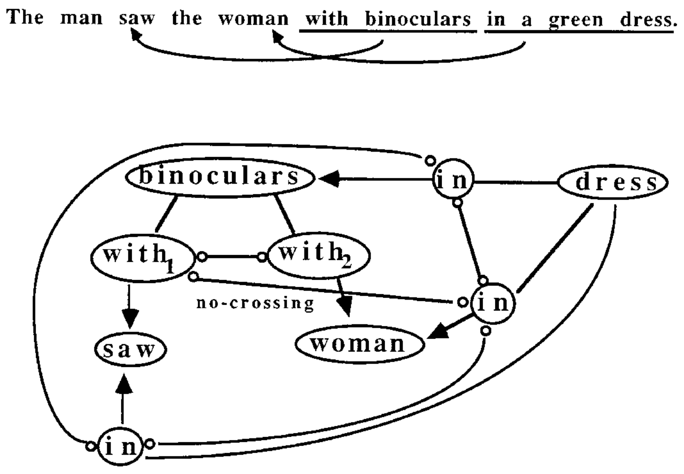

Figure 5: The No-Crossing-of-Branches Rule.

Each attachment node is associated with a unique prepositional phrase and a unique attachment point. By consulting the time stamps for each pair of constituents affiliated with an attachment node, we can specify a time interval which describes the scope of this particular attachment node. The longer the time interval, the more distance we have between the prepositional phrase and its potential attachment point.

Assume two attachment nodes are given the time intervals [a,b] and [c,d], where $a < b$, $c < d$, and $a < c$. Then these two attachment nodes will violate the no-crossing rule if and only if $c < b < d$. Under no other condition will we see a violation of the no-crossing rule. It is easy to check the pairwise combinations of attachment nodes in our network to determine if any pairs qualify as a violation of the rule. If we find any such pairs, we can then incorporate this information into our lateral inhibition network by adding an inhibitory link between the two offending attachment nodes to indicate that these nodes are mutually exclusive and incompatible with one another.

This inhibitory link will operate as a soft constraint in the sense that a stable network could conceivably settle out with maximal activation on two attachment nodes that cross. In such a case we would simply accept the sentence interpretation that violated the no-crossing constraint, presumably because semantic considerations overwhelmed the no-crossing rule. In any case, the no-crossing rule is implemented using the same data-driven technique for soft-constraints that we use to arbitrate competing word senses and attachment points. Syntactic constraints and semantic preferences then operate within the relaxation algorithm on equal footing, without any opportunity for one constraint type to dominate over the other. The various constraints reinforce or compete with one another in a truly heterogeneous manner until a globally optimal interpretation of word senses and pp-attachments is obtained.

Most sentence analyzers which implement the no-crossing rule recognize it as a hard constraint and cannot balance it against semantic considerations as we can with CIRCUS. Since people are willing to violate the no-crossing rule some of the time, but not all the time, it seems obvious that a flexible mechanism is needed to utilize the no-crossing constraint. We believe that numerical relaxation is extremely convenient for problems of this type. Moreover, any purely symbolic technique for the no-crossing rule will be necessarily brittle and arbitrary in trying to balance syntactic sensitivities against semantic preferences. The subsymbolic method of network relaxation is far better suited to the resolution of multiple soft constraints.

0.5 Underlying Memory Models

Our overview of CIRCUS has been dominated by descriptions of control structures. We have described interactions between syntactic analysis and the marker passing algorithm, as well as interactions between the marker passing algorithm and numerical relaxation. We have seen how to construct appropriate networks for both marker passing and numerical relaxation, and we have looked at specific sentences to illustrate all of these ideas. Up until now, we have managed to finesse as much as possible the question of underlying memory structures, although we have explained where memory interactions are needed.

There are three basic types of memory that drive the sentence analysis of CIRCUS. They are:

(1) A Lexical Dictionary (LD)
(2) Structured Relational Preferences (SRP)
(3) Unstructured Relational Preferences (URP)

The LD is used during syntactic analysis and network construction for the marker passing algorithm, SRP are used during marker passing, and URP are used for numerical relaxation. We will describe the role played by each form of memory, and indicate where we stand with respect to our current implementation of CIRCUS.

Of the three memory modules, the LD is the most straightforward and least interesting aspect of memory. Here is where we set up associations between words and word senses, word senses and case frame definitions, word senses and parts of speech. Syntactic predictions are associated with any word sense that assumes a semantic case frame interpretation. In general, there must be at least as many syntactic predictions for a given word sense as there are case frame slots for that word sense.

The machinery for handling dictionary information is fully implemented in CIRCUS and completely adequate for the sorts of sentences we've been discussing in this paper as well as others.[5] We do not mean to suggest that there are no interesting problems in designing LD definitions. One must always make hard decisions about primitive decomposition within a case frame representation, where to draw the line between two distinct word senses, and whether or not a sentence representation should strive to be free of inferences. But all of these problems are primarily problems in representation rather than memory per se. So let us leave the LD and concentrate on the remaining two memory structures needed to drive CIRCUS.

0.5.1 Structured Relational Preferences

Structured Relational Preferences (SRP) refer to constraints that aid in the identification of case frame slot fillers. These are generally soft constraints, since they can be violated if syntactic considerations force a case frame instantiation that contradicts normal semantic expectations. Most predictive case-frame

[5]For example, CIRCUS can simulate garden path effects for reduced relative clause sentences by tracking competing case frame instantiations (via concept node activation levels) within PSM.

parsers utilize some form of a slot constraint mechanism, and the standard solution involves semantic feature checking. Indeed, simple semantic features are what CIRCUS is currently using as well.

Each word sense in the LD is associated with a semantic feature vector, and the feature nodes in PSM normally do nothing more than check for the presence of a given semantic feature. We could improve on the current solution a bit by introducing an inheritance tree to organize our semantic features, but there are other problems with semantic features that transcend the question of how cleverly we organize them.

For example, consider the problem of instantiating a case frame for eating. "John ate a hamburger," is fine as long as John is a human, but "John ate some hay," should signal a semantic violation. Then again, eating hay is fine if you're a horse, so we can't hope to nail the preference on the basis of the object slot filler alone. Here we need to consider both the agent and the object together in order to determine whether or not the object qualifies as food for that particular agent. A simple semantic feature called "food" is not sufficient since the agent doing the eating must be taken into consideration as well.

At this point we have to decide whether or not we are primarily interested in building a practical system for some limited domain, or a system that makes theoretical claims about psychological validity. For the sake of building practical systems, simple semantic features are largely adequate. But if we want to make claims about the correct model for SRP in general, we are more inclined to invoke a model of episodic memory structuring. There are a variety of possibilities here, but they must all incorporate some capability for learning from experience. Some recent attempts in this direction have made tentative steps toward outlining an episodic model of SRP, including the DMAP system (Riesbeck & Martin, 1986), and ELAN (Lehnert, 1987b). We can also imagine a connectionist strategy using a learning technique like back propagation and distributed representations, so we see an opportunity here for either symbolic or subsymbolic SRP models.

By moving away from semantic discriminations into episodic ones, we can set up preferences that may be extremely specific, but nevertheless valid. For example, if we know a particular individual named John, and we also know that John is a vegetarian, we should be concerned to hear that John is eating a hamburger. This constitutes a violation of episodic expectations. If our experience never goes beyond knowledge about one person named John, SRP may effectively take John to be a good representative of all humans. The issue of dynamic generalizations and appropriate levels of abstraction become important for any SRP model that operates on the basis of episodic memory. It necessarily follows that any episodic model of SRP must incorporate a model of similarity-based (inductive) learning as well. The earliest strategy for handling SRP as a dynamic episodic memory structure was illustrated by the IPP system which read stories about terrorism and created generalizations based on those stories (Lebowitz, 1983a,b).

At the present time, connectionist models for inductive learning are currently limited to simple associations or a single relational predicate with two arguments. If a connectionist network could scale up to an arbitrary set of relational predicates with arbitrarily many arguments, we would have the basis for an episodic memory that yields case frame slot-filling preferences. SRP therefore provides us with an opportunity to experiment with high-level connectionism, but the demands of SRP seem to push techniques like back propagation beyond their reasonable limits in terms of training requirements.

0.5.2 Unstructured Relational Predicates

Unstructured Relational Predicates (URP) are needed by the relaxation algorithm in CIRCUS to provide initial activation levels for relational nodes representing pp-attachments. Although a relational predicate is, in fact, a structured entity, we are calling the memory behind these predicates unstructured because the preferences needed to handle relational node initialization are better described in terms of semantic knowledge instead of episodic knowledge. At some early stage of development, a child might answer the question "Can a person be in a car?" by thinking of specific people getting into a specific car, but this type of knowledge must generalize very quickly to levels of abstraction that drop their episodic origins. In general, we will assume that event-oriented memory remains essentially episodic, while relational memory quickly evolves into semantic memory organization even if its earliest roots are episodic. These assumptions deserve substantial discussion, but we will not digress here to pursue it.

Since data-driven preference semantics requires knowledge about relational constraints, URP is properly addressed by semantic memory models rather than episodic ones. In section 4.1 we saw how CIRCUS could invoke a simplistic conditional statement for the purpose of initializing our relaxation network. For limited applications, this primitive notion of URP might prove to be adequate. However, any extensive application that requires data-driven semantics would probably encounter a difficult obstacle if we have to scale-up by tuning these conditionals in order to handle a large class of attachment interactions.

To address this issue of scaling up, we have been investigating the utility of back propagation as a technique for training a semantic memory model to provide an URP capability. Using examples from a corpus of scientific papers, we've trained networks by presenting triples representing noun-preposition-noun combinations (e.g. "temperature in Fahrenheit") along with one of two possible outcome values: "plausible" or "implausible." Each network consists of 32 input units (16 binary features for each noun), 12 hidden units, and one output unit. We train a separate network for each preposition, but the binary features used for encoding input nouns remain fixed across all networks. Once training is completed, we can produce correct plausibility judgments for 94% of the training corpus, and 81% of a novel test set. A more detailed account of this experiment can be found in (Wermter, 1989).

Although our hit rate on novel test items may not appear to be overwhelmingly impressive, it is important to note that the novel test set contains nouns which were not necessarily present in the training corpus. To add a new noun to URP, we need only structure its representation in terms of the 16-feature vector. No additional knowledge is required since the backprop network has already encoded the necessary knowledge for making relational plausibility judgments. Substantial training is required to add a new preposition to URP, but in general, we can expect to have more nouns than prepositions, so this requirement does not seem unreasonable. If we can improve the hit rate on novel test items by either restructuring our feature vector or opting for more training, it appears that back propagation may be an effective strategy for scaling up substantial URP memories.

0.6 Conclusions

In this paper we have concentrated on the basic architecture of CIRCUS in an effort to show how (1) stack-oriented syntactic analysis, (2) marker passing for predictive preference semantics, and (3) numerical relaxation for data-driven preference semantics, can operate together in order to take full advantage of each processing strategy. We have also seen how one strictly syntactic constraint is easily handled by the numeric relaxation algorithm, suggesting that syntactic and semantic concerns are not always best served by separate processing strategies (despite our general tendency with CIRCUS to do just that). It might be better to characterize the three modules of CIRCUS in terms of (1) local syntactic constraints, (2) local semantic constraints, and (3) global constraints of either a syntactic or semantic nature. This somewhat more accurate portrayal of the CIRCUS processing modules suggests a useful pattern for understanding the respective roles of symbolic and subsymbolic information processing.

In section 4.2 we talked about the top-down/bottom-up relationship between predictive preference semantics and data-driven preference semantics. We further pointed out that the top-down component of predictive preference semantics was symbolic and serial, while the bottom-up component of data-driven preference semantics was subsymbolic and parallel. Now we can also add a local/global edge to this same boundary line, so the following dichotomies emerge:

predictive	vs.	data-driven
top-down	vs.	bottom-up
symbolic	vs.	subsymbolic
serial	vs.	parallel
local	vs.	global

These contrasting dichotomies are familiar to anyone who has been watching the AI/connectionist debates in recent years, but it is unusual to find a single system where both aspects of each dichotomy are present and operating in concert. In general, AI models tend to be situated in the left column, while connectionist models are more comfortable with the right side. CIRCUS represents a powerful synthesis of symbolic and subsymbolic techniques, utilizing properties from both columns as needed to handle a variety of problems.

Inevitably, people like to understand the claims associated with system implementations. Is the model being proposed psychologically valid, physiologically motivated, or just a clever hack? More importantly, for any claim being made, which parts of the system implementation are meant to be taken seriously, and which should be dismissed as necessary kludges?

In trying to answer these questions about CIRCUS, we must first admit that the exploratory aspect of CIRCUS is a synthesis of many ideas – some new and some old. We were primarily interested in designing a conceptual sentence analyzer that did a few things better than other conceptual sentence analyzers. Ideas were pulled from a variety of places, including the connectionist literature on sentence analysis (especially Waltz & Pollack, 1985), but there was no one theoretical motivation behind the design of CIRCUS. It was, and remains, largely an engineering effort. As such, we believe that CIRCUS has been very successful insofar as it is currently being used to provide natural language processing capabilities for other research projects (Wermter, 1989; Lehnert, et. al., 1989).

On the other hand, CIRCUS is fully consistent with a number of theoretical claims. Some of these claims are specific to theories about natural language processing, while others are more general in nature. To researchers intent on the pursuit of "high-level connectionism," we are in a position to make one very general claim about the application of subsymbolic techniques to processes associated with "high-level" cognition. Namely, there is no reason to believe that a single information processing mechanism can be held accountable for processes as complex as sentence analysis.

Indeed, we have seen many attempts to reduce sentence analysis to a single computational mechanism fail time and time again. The list is long and thoroughly researched: augmented transition networks, semantic grammars, chart parsers, production systems, expectation-based systems, and more recently, marker passing, back propagation, and spreading activation algorithms. In each case, we see a mechanism with obvious shortcomings and apparently insurmountable problems. In each case, one or two aspects of the problem at hand are addressed effectively and attractively, while everything else remains beyond the scope of the current model. And still the hope lives on that someday we will manage to find the one correct mechanism. This desire to reduce a complex phenomenon to a simple explanation is understandable from the perspectives of elegant science and aesthetic sensibility, but it should not preclude from consideration the possibility that such explanations are simply not in the cards.

Cognitive processes must not be confused with the phenomena of physics or other natural sciences. Cognition is a biological entity which has evolved over a long period of time as an adaptive mechanism. Any system that takes a million years to develop is not likely to be elegant or optimal. On the contrary, we have every reason to believe that there are multiple layers of information processing mechanisms, each contributing to specific aspects of the problem at hand, much as the gross anatomy of the brain layers newer structures over older ones.

At the workshop where CIRCUS was first presented, one participant described CIRCUS as an example of a "grandiose architecture." The import of this characterization seemed clear: no one prefers to go after solutions of this complexity - surely we can do better by persisting in the search for the (one) right mechanism. And indeed, there are lessons to be learned by pushing a single mechanism as far as it will go. But many of these lessons are clearly at odds with the idea of a single processing mechanism. Some aspects of language processing are characterized by memory limitations and serial processing effects (limited syntactic embedding, garden-pathing on certain reduced-relative clauses, the general inability to handle sentences of arbitrary syntactic complexity), while other effects can only be explained in terms of parallel processing (lexical priming phenomena are the most compelling examples here). If we take all the available evidence into consideration, the hope for a single explanatory mechanism seems truly quixotic.

Moreover, we have no reason to feel defeated by solutions that require "grandiose architectures." On the contrary, any computational model of natural language processing that remains extensible and consistent with human information processing capabilities must be welcomed, even if some aesthetic sensibilities need to be sacrificed along the way. So CIRCUS is presented here without apology, but with some defensiveness. We are not too proud to propose a grandiose architecture for natural language processing: natural language is a complicated problem. When we have competing explanations of equal power and breadth, we will happily invoke Occam's razor and allow simplicity to dominate. Until then, it is better to worry about extensibility first, and aesthetics later. If "high-level connectionism" is going to flourish and succeed, this shift in research priorities may be a crucial first step.

0.7 References

Birnbaum, L. & Selfridge, M. (1981). Conceptual Analysis of Natural Language. In R.C. Schank & C. Riesbeck, (Eds.), *Inside Computer Understanding*, pp. 318-353. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Cullingford, R. (1986). *Natural Language Processing*, Totowa, New Jersey: Rowman & Littlefield.

Dyer, M. (1983). *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension.* Cambridge, MA: MIT Press.

Feldman, J.A. and Ballard, D.H. (1982). Connectionist Models and Their Properties. *Cognitive Science* (Vol. 6, no. 3. pp. 205-254).

Ford, M., Bresnan, J. & Kaplan, R. (1981). A Competence-Based Theory of Syntactic Closure. In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations.* Cambridge, MA: MIT Press.

Frazier, L. & Fodor, J. (1979). The Sausage Machine: A New Two-State Parsing Model. *Cognition* (Vol. 6, pp. 191-325).

Lebowitz, M. (1983a). Memory-Based Parsing. *Journal of Artificial Intelligence*, (Vol. 21, No. 4, pp. 363-404.).

Lebowitz, M. (1983b). Generalization From Natural Language Text. *Cognitive Science*, (Vol. 7, No. 1, pp. 1-40).

Lehnert, W., Cardie, C., Riloff, E., Swaminathan, K. &. Wermter, S. (1989). Knowledge Acquisition from Research Documents. Submitted to the Eleventh International Joint Conference on Artificial Intelligence.

Lehnert, W.G. (1987a). Case-Based Problem Solving with a Large Knowledge Base of Learned Cases. *Proceedings of the Sixth National Conference on Artificial Intelligence* Seattle, WA.

Lehnert, W.G. (1987b). Learning to Integrate Syntax and Semantics. In *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA.

Lytinen, S. (1984). Frame Selection in Parsing. In *Proceedings of the National Conference on Artificial Intelligence.* Austin, Texas.

Riesbeck, C. (1975). Conceptual Analysis. In R.C. Schank, (Ed.) *Conceptual Information Processing.* Amsterdam: North Holland.

Riesbeck, C. & Martin, C. (1986). Direct Memory Access Parsing. In C. Riesbeck and J. Kolodner (Eds.), *Experience, Memory and Reasoning*, Hillsdale, NJ: Lawrence Erlbaum.

Riesbeck, C., & Schank, R. (1976). Expectation-Based Analysis of Sentences in Context. Research Report #78. New Haven, CT: Department of Computer Science, Yale University.

Schank, R.D. (Ed.) (1975). *Conceptual Information Processing.* Amsterdam: North Holland.

Schank, R.D. and Riesbeck, C. (1981). *Inside Computer Understanding: Five Programs Plus Miniatures.* Hillsdale, NJ: Lawrence Erlbaum.

Swinney, D. (1884). Theoretical and Methodological Issues in Cognitive Science: A Psycholinguistic Perspective. In Kintsch, Miller & Polson (Eds.), *Method and Tactics in Cognitive Science.*

Tait, J.I. (1983). Semantics-directed Parsing. In K. Sparck Jones & Y. Wilks (Eds.), *Automatic Natural Language Parsing* (pp. 169-177). New York, NY: John Wiley & Sons.

Waltz, D., & Pollack, J. (1985). Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. In *Cognitive Science*, (Vol 9, No. 1).

Wermter, S. (1989). Integration of Semantic and Syntactic Constraints for Structural Noun Phrase Disambuation. Submitted to the Eleventh International Joint Conference on Artificial Intelligence.

Wilks, Y., Huang, X. & Fass, D. (1885). Syntax, Preference and Right Attachment. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, (pp. 779-784).